Discrete Applied Mathematics 159 (2011) 812–825

Contents lists available at ScienceDirect

Discrete Applied Mathematics

journal homepage: www.elsevier.com/locate/dam

A direct barter model for course add/drop process

Ali Haydar Özer, Can Özturan *

Department of Computer Engineering, Boğaziçi University, 34342, Bebek, Istanbul, Turkey

ARTICLE INFO

Article history:
Received 31 July 2009
Received in revised form 4 January 2011
Accepted 6 January 2011
Available online 5 February 2011

Keywords:
Add drop
Student registration
Timetabling
Course scheduling
Bartering
Barter network
Network flow

ABSTRACT

Even though course timetabling and student scheduling problems have been studied extensively, not much has been done for the optimization of student add/drop requests after the initial registration period. Add/drop registrations are usually processed with a first come first served policy. This, however, can introduce inefficiencies and dead-locks resulting in add/drop requests that are not satisfied even though they can, in fact, be satisfied. We model the course add/drop process as a direct bartering problem in which add/drop requests appear as bids. We formulate the resulting problem as an integer linear program. We show that our problem can be solved polynomially as a minimum cost flow network problem. In our model, we also introduce a two-level weighting system that enables students to express priorities among their requests. We demonstrate improvement in the satisfaction of students over the currently used model and also the fast performance of our algorithms on various test cases based on real-life registration data of our university.

1. Introduction

In universities, course timetabling (CT), student scheduling (SS) and add/drop processes involve the coordination of various resources and entities. CT basically deals with the allocation of time slots and classrooms to courses by taking into consideration issues such as preferences of instructors and classroom locations. Given a timetable, in SS phase, students select courses according to their needs and preferences. Because of course and section quota restrictions or enrollment balancing requirements among the sections, it is not possible to satisfy the needs and preferences of all the students. Therefore, some policy or algorithm needs to be employed in SS phase for the assignment of students to courses and sections. During the add/drop phase, a readjustment of the assignment solution in SS phase basically takes place by the addition, dropping and swapping of courses and/or sections. In the literature, phases CT and SS have been extensively studied (see, for example, surveys [6,7,20,27]). Some approaches tackled either CT or SS exclusively. Some approaches coupled these two phases and solved the combined course timetabling and student scheduling problem. In this paper, our focus will be on the add/drop process. Not much has been done for this phase—we are aware of only one work (that of Graves et al.'s [14]) that addresses the add/drop process. The add/drop process has an important difference from that of CT and SS. A student may have been already assigned to a seat in a course or section from SS phase and he may want to swap (barter) this seat that he owns with another seat owned by other students in another course or section. Hence, one can say that whereas CT and SS phases can be modeled as an assignment problem, for add/drop process bartering is a more appropriate model.

We were motivated to develop a direct barter model for the add/drop process because of some problems we noticed during add/drop periods at our Boğaziçi University. Since 1998, web based online registration system has been used for course registration [5]. Before the beginning of each semester, students are admitted to the system and are allowed to take courses if both prerequisites of the courses are satisfied and the quotas of the courses permit. The system works on a first

* Corresponding author. Tel.: +90 212 359 7225; fax: +90 212 287 2461.
E-mail addresses: ozeralih@boun.edu.tr, ahozer@gmail.com (A.H. Özer), ozturaca@boun.edu.tr (C. Özturan).

0166-218X/$ – see front matter
doi:10.1016/j.dam.2011.01.004

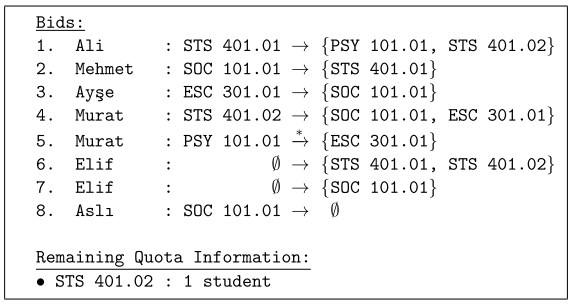

Fig. 1. Example problem for illustrating add, drop, and barter bids.

come first served (FCFS) policy basis and at the beginning of each registration period, a race occurs among students for popular courses. Generally, the quotas of the popular courses are filled within the first few hours of online registration period. After the registration period, the semester begins and during the first week of the semester, the students attend and evaluate their courses. At the end of this week, add/drop period of one week begins and the students are allowed to change their courses and/or sections of their courses. Because of the FCFS basis of the system and the quota restrictions, when a student drops a course, he may not be able to take it again. This situation forces a student who wants to change his course, to first try to add a new course, and then drop the old course. Although this does not pose a problem if the quotas of the courses are not full, it does pose a problem for the popular courses. It is observed in Boğaziçi University student registration system that the current FCFS based system causes deadlock situations, and hence reduces the total satisfaction of students. Although different implementations of FCFS approach exist in different registration systems, all FCFS based systems are prone to the same problem. For instance, in UniTime [21,29], which is an open source enterprise system for automated construction of course timetables and student schedules, when a student wants to add a course which is not available, the student is assigned to the wait-list of that course. Wait-lists are processed automatically in FCFS manner as one seat becomes available for the corresponding course. Therefore, since a student who wants to change his course cannot be sure whether he would be assigned to the new course, he would not want to drop the course he has already assigned until he obtains a seat in the new course. Thus, this would also lead to the same problem.

In order to increase the efficiency of add/drop process compared to the current FCFS based system of our university, a direct barter model for the course add/drop process is proposed. The objective of the model is to increase the total satisfaction of students while preserving fairness among them. For this purpose, along with the usual add and drop requests, this model allows students to barter the courses they want to drop for the courses they want to add. Students express their requests through submitting multiple add, drop and barter bids and in each add and barter bid, they can declare a set of alternative courses to be added. Besides, in this model, they can indicate relative priorities of their bids and the courses they want to register for. For instance, if a student prefers course A over course B, and course B over course C, he just declares $A \succ B \succ C$. Furthermore, students can request the same course or the same set of courses in multiple bids and can also declare restriction sets in which only one course can be added to the schedule.

In this paper, we contribute a formal development of the model. We present a network flow based algorithm that allows us to solve the problems in strongly polynomial time. We also compare the solutions of our model with that of the FCFS approach based on real-world student registration data and present the performance of our algorithms on various tests.

In the next section, we present an example with which we explain our model for the course add/drop process. In Section 3, we formally define and formulate our model using integer programming. Then, in Section 4 we present a minimum cost network flow solution of our problem and in Section 5, we present the experimental results. A review of the related literature is given in Section 6. Finally, the paper is concluded in Section 7.

2. A motivational example and the model

In this section we present an example scenario for add/drop process on which we explain our direct barter model. Assume that during the registration period, students Ali, Mehmet, Ayşe and Aslı have been registered for courses STS 401.01, SOC 101.01, ESC 301.01 and SOC 101.01 respectively. Murat, on the other hand, has been registered for both STS 401.02 and PSY 101.01. Suppose that during the add/drop period, the students declare *add*, *drop*, and *barter bids* as shown in Fig. 1.

Bids 1–5 are examples of a barter bid. In a barter bid, the left hand side of the arrow indicates the course to be dropped and the right hand side indicates the course to be added. A barter bid as the name suggests enforces the student to drop the course on the left hand side if he adds the new course on the right hand side. For instance, in bid 3 Ayşe wants to drop ESC 301.01 if she could add SOC 101.01 to her course list. Bids 6 and 7 are examples of an add bid. An add bid states that the student wants to add the course on the right hand side without dropping any other course. Likewise, a drop bid, e.g. bid 8, states that the student wants to drop the course on the left hand side without adding any other course.

Bids 1, 4 and 6 are different from the others in terms of having a *request set* of more than one course on the right hand side. These bids are called *multi-bids*. A *multi-barter bid* states that the student is indifferent, at least to some degree, to the

set of requested courses and he is willing to drop the course on the left hand side if he could add *any one* of the courses in this set. Similarly, a *multi-add bid* states that the student wants to add any one of the courses in this set without dropping any other course. Since drop bids are not restricted with quota constraints, they should always be satisfied. Therefore, we do not need to explicitly incorporate multi-drop bids in the model.

Multi-bids can be considered as combinations of two or more single bids which are XOR'ed. For instance, in bid 1, Ali wants to add either PSY 101.01 or STS 401.02 but not both and drop STS 401.01 on the condition that his add request is satisfied. This bid can be represented as a combination of two XOR'ed bids, STS 401.01 →{PSY 101.01} and STS 401.01 → {STS 401.02}. Bid 1 is satisfied if exactly one of these bids is satisfied. By introducing multi-barter bids, without losing generality, we can now safely assume that there are *no two barter bids* of a student that have the same course on the left hand side since such bids can be combined and represented as one multi-barter bid. In addition to the multi-barter bid mechanism, the model allows a student to mark a barter bid (either single or multi-barter) as *drop-unless-barter* meaning that the student wants to drop the course on the left hand side if bartering of this course for another course in the request set is not possible. In the given example, it is indicated using a star above the arrow of the bid 5. In this bid, Murat wants to barter PSY 101.01 for ESC 301.01 and if ESC 301.01 cannot be added, Murat wants to drop PSY 101.01. Again, by further introducing drop-unless-barter mechanism, drop and barter requests can be combined and again without losing generality, we can state that there cannot be *any two bids* of a student that have the same course on the left hand side.

The add/drop process based on the direct barter model is a batch process and consists of two phases, a *bid submission phase* in which the students are allowed to submit bids to the system or retract bids from the system and a *solution phase* in which the optimum solution is calculated. Depending on the duration of the add/drop period, these phases can be repeated as many times as necessary. For instance, for each day of the add/drop period, the bids can be collected from the students throughout the day and the solution can be calculated at the end of the day and announced to the students afterwards.

Expressing preferences—weighted model

Although the described unweighted model helps to increase the total satisfaction of students, it can further be improved by assigning weights to the bids. In the weighted model, add, drop, and barter bids are defined respectively as follows:

$$c_\emptyset \xrightarrow{w_i} \{(a_{i1}, w_{i1}), (a_{i2}, w_{i2}), \ldots, (a_{ip}, w_{ip})\}$$
$$d_i \xrightarrow{w_i} \{(c_\emptyset, 0)\}$$
$$d_i \xrightarrow{w_i} \{(a_{i1}, w_{i1}), (a_{i2}, w_{i2}), \ldots, (a_{ip}, w_{ip})\}$$

where i is the index of the bid, w_i is the weight of the bid i, and d_i is the course to be dropped. $c_\emptyset$ denotes the *null course* used for representing the course to be dropped or added for add and drop bids respectively. In the weighted model, weights are assigned not only to the bids, but also to the requested courses. Therefore, the request set contains tuples (a_{ij}, w_{ij}) where a_{ij} is the requested course, w_{ij} is the associated weight, and p is the number of the requested courses.

By assigning weights to the bids and the requested courses, the model becomes more powerful in the sense that it enables students to express their preferences inside the bids. For instance, weight values can be assigned to a student's bids indicating the degree of his preferences for his bids. The weight of the most favored bid would be the highest and the least favored would be the lowest. Likewise, for each multi-bid, weight values can also be assigned to the requested courses in the request set if he is not totally indifferent to these courses. Considering the quota restrictions and the submitted bids, among the possible courses in the request set, the one with the highest weight would be added to the student's course list. Besides the ability to express preferences among the bids and the requested courses, the weighted model also enables favoring some students over others. Special students such as graduating students can also be favored officially by their department by increasing or maximizing the weights of their bids. This will ensure that if the quotas of the courses are available then these students will be the first to add the courses they want. Similarly, using the same mechanism, successful students, i.e. the students with higher grade point average (GPA), can also be favored depending on the policy of the university.

3. Formulation of the model

The weighted direct barter model is formally defined as follows: let $C = \{c_1, c_2, \ldots, c_m\}$ be the set of m courses and $Q = (q_{c_1}, q_{c_2}, \ldots, q_{c_m})$ be the tuple of remaining quotas where q_{c_k} is the remaining quota of course c_k ($1 \leq k \leq m$, $q_{c_k} \in \mathbb{Z}^+ \cup \{0\}$). Let $S = \{s_1, s_2, \ldots, s_t\}$ be the set of t students. We define B_l as the set of bids submitted by a student s_l and the set of all bids, B, is defined as $B = \bigcup_{l=1}^{t} B_l$. Each bid is denoted by a triplet, $b_i = (d_i, w_i, R_i)$, where d_i is the course to be dropped for *barter* and *drop* bids or the null course, $c_\emptyset$, for *add* bids ($d_i \in C \cup \{c_\emptyset\}$), $w_i \in \mathbb{R}^+$ is the weight and R_i is the request set of the bid b_i. The request set of a bid is either $\{(c_\emptyset, 0)\}$ for *drop* bids or a set of two tuples, $R_i = \{(a_{i1}, w_{i1}), (a_{i2}, w_{i2}), \ldots, (a_{ip}, w_{ip})\}$, for *barter* and *add* bids. Each tuple (a_{ij}, w_{ij}) in R_i indicates the requested course, that is the course to be added, and the associated weight respectively ($a_{ij} \in C$, $w_{ij} \in \mathbb{R}^+$). Finally, the set $D \subseteq B$ denotes the bids which are marked as *drop-unless-barter*.

A bid b_i is called *satisfiable* if at least one of the courses in the request set has one or more remaining quota or there exists at least one satisfiable bid whose course to be dropped is in the request set of b_i. Formally, given $b_i = (d_i, w_i, R_i)$ and $b_l = (d_l, w_l, R_l)$ the following predicate is true: $\forall i$ (satisfiable $(b_i) \Leftrightarrow \exists j ((a_{ij}, w_{ij}) \in R_i \wedge ((q_{a_{ij}} \in Q \wedge q_{a_{ij}} > 0) \vee \exists l$ (satisfiable $(b_l) \wedge d_l = a_{ij}))))$.

By the definition, all drop bids are satisfiable. The objective of the model is to find the set of satisfiable bids that maximizes the sum of the weights, that is the sum of both bid weights and weights of the requested courses, and hence the total satisfaction of students.

In order to formulate the model using integer programming, two binary variables are introduced. The binary decision variable x determines the satisfied bids and y determines the requested course which is added if the corresponding bid is satisfied. Formally,

$$x_i = \begin{cases} 1, & \text{if bid } i \text{ is satisfied} \\ 0, & \text{otherwise} \end{cases} \quad \text{and} \quad y_{ij} = \begin{cases} 1, & \text{if course } a_{ij} \text{ is added for bid } i \\ 0, & \text{otherwise.} \end{cases}$$

It should be noted that for a drop-unless-barter bid b_i, the meaning of satisfaction is slightly different. $x_i = 0$ means that the student drops the course without adding any other course and $x_i = 1$ means that the student barters the course for another course. The integer programming formulation of the model is as follows:

$$\text{maximize} \quad \sum_{\forall i | b_i \in B} \left(w_i x_i + \alpha \cdot \sum_{\forall j | (a_{ij}, w_{ij}) \in R_i} w_{ij} y_{ij} \right) \tag{1}$$

$$\text{subject to} \quad \sum_{\forall i | b_i \in (B \setminus D) \wedge d_i = c_k} x_i - \sum_{\forall i,j | b_i \in B \wedge (a_{ij}, w_{ij}) \in R_i \wedge a_{ij} = c_k} y_{ij} \geq -\left(q_{c_k} + \sum_{\forall i | b_i \in D \wedge d_i = c_k} 1 \right) \quad (\forall k \mid c_k \in C) \tag{2}$$

$$x_i - \sum_{\forall j | (a_{ij}, w_{ij}) \in R_i} y_{ij} = 0 \quad (\forall i \mid b_i \in B \wedge R_i \neq \{(c_\emptyset, 0)\}) \tag{3}$$

$$x_i = 1 \quad (\forall i \mid b_i \in B \wedge R_i = \{(c_\emptyset, 0)\}) \tag{4}$$

$$\sum_{\forall i,j | b_i \in B_l \wedge (a_{ij}, w_{ij}) \in R_i \wedge a_{ij} = c_k} y_{ij} \leq 1 \quad (\forall k, l \mid c_k \in C \wedge s_l \in S) \tag{5}$$

$$x_i, y_{ij} \in \{0, 1\} \quad (\forall i, j). \tag{6}$$

Note that in this formulation, the objective line in Eq. (1) maximizes the sum of the bid weights and the weights of the courses in the request sets. In this equation, α factor is a constant positive number to be determined according to the actual weight values and the number of bids which is described in the next section. Eq. (2) enforces quota restrictions of the courses. For each course, the number of students dropping the course (including the drop and drop-unless-barter bids) plus the remaining quota of the course should be greater than or equal to the number of students who added the course. Eq. (3) expresses the satisfaction criterion: an add bid or a barter bid is satisfied if exactly one of the courses in the request set is added. Eq. (4) ensures that all the drop bids are satisfied. Finally, Eq. (5) prevents students from adding the same course to their schedule more than once.

3.1. Determining the weight values and the α factor

The main objective of the direct barter model is to increase the total satisfaction of students while preserving fairness among them. In this section, we propose a method for defining the parameters of the model, that are the weights of the bids, w_i, the weights of the requested courses, w_{ij}, and the α factor, in accordance with this objective.

In this method, each student, s_l, is responsible for ranking his bids according to his preference instead of defining the actual weights of the bids. Based on this ranking, he constructs his *preference list*, a permutation of his bids sorted in descending order of his preference. Then, this preference list is used as B_l, the set of bids submitted by the student s_l. So, in the set $B_l = \{b_l^{(1)}, b_l^{(2)}, \ldots, b_l^{(u)}\}$, the bid $b_l^{(1)}$ is the most preferred bid with highest rank number of 1 and the bid $b_l^{(u)}$ is the least preferred bid with the lowest rank number of u (i.e. $\forall l : b_l^{(1)} \succ b_l^{(2)} \succ \ldots \succ b_l^{(u)}$). Note that, for this definition we use a different indexing scheme for referring the bids in the set B_l in order to prevent confusion with the indexing scheme for referring the bids in the bid set B. Subscript of a bid denotes the owner of the bid and the superscript, which is the index number in the set B_l, denotes the rank of the bid. We also denote the weight of a bid $b_l^{(r)}$ with $w_l^{(r)}$. Given the ordered set of bids B_l of the student s_l, the weight of a bid $b_l^{(r)}$ is defined as follows:

$$w_l^{(r)} = 2^{h-r} \quad (\forall l, r \mid b_l^{(r)} \in B_l). \tag{7}$$

In this function, h is the index of the lowest ranked bid among all the bids, and therefore the minimum bid weight value, $w_{\min}$ is 1. This function ensures that the weights of the bids that have the same rank among the students are equal and for each bid $b_l^{(r)} \in B_l$, the weight of the bid is greater than the sum of the weights of the lower ranked bids in the same set. Therefore, this bid weight function enables the model to satisfy as many higher ranked bids as possible while preserving fairness among the students.

Defining the weight values for the requested courses is straightforward. As for the bids, each student declares the courses in the request set such that the more preferred course comes before the less preferred course (i.e. $\forall i \mid b_i \in B : a_{i1} \succ a_{i2} \succ \cdots \succ a_{ip}$). Since for each bid at most one requested course can be added to the students schedule in the optimum solution, the only requirement for weights of the requested courses is to ensure that the more preferred requested course has higher weight value than that of the less preferred course. Therefore, the weight value of a requested course is simply defined as:

$$w_{ij} = m - j + 1 \quad (\forall i \mid b_i \in B) \tag{8}$$

where m is the number of courses ($m = |C|$). This simple function ensures that the requested courses with the same rank have equal positive weights among all the bids.

As seen from the objective function given in Eq. (1), there are two objectives of the model: the first objective is to maximize the number of satisfied bids according to bid weight values and the second objective is for each satisfied bid to add one requested course with the maximum possible weight. In order to maximize the total satisfaction of students, the first objective is favored against the second objective so that when finding the optimum solution among the feasible solutions, the solution with the maximum sum of the bid weights is chosen as the optimum. However, if there are multiple solutions with the same maximum sum of the bid weights, then the solution with the maximum sum of the weights of the requested courses is chosen among these solutions. In order to provide this feature, the ranges of these two types of weights should be separated in order to cancel the effects of the latter to the former. For this purpose, a constant factor α for scaling the sum of the weights of the requested courses is introduced in the objective function. The α factor is defined as follows:

$$\alpha = \frac{1}{(|B| - 1) \cdot m}. \tag{9}$$

The following proposition proves that using these weight functions and the α value, the first objective of the model is favored against the second objective.

Proposition 1. *Given any two different feasible solutions with different sums of the bid weights for the direct barter model, the solution with higher sum of the bid weights has higher objective value according to Eq. (1) independent of weights of the requested courses.*

Proof. Let B_1 and B_2 be two sets of satisfiable bids that correspond to any two feasible solutions for the direct barter model, and z_1 and z_2 be the corresponding objective values such that

$$z_1 = \sum_{\forall i \mid b_i \in B_1} \left(w_i + \alpha \sum_{\forall j \mid (a_{ij}, w_{ij}) \in R_i \wedge y_{ij} = 1} w_{ij} \right) \quad \text{and} \quad z_2 = \sum_{\forall i \mid b_i \in B_2} \left(w_i + \alpha \sum_{\forall j \mid (a_{ij}, w_{ij}) \in R_i \wedge y_{ij} = 1} w_{ij} \right). \tag{10}$$

We will show that if the sum of the bid weights of the B_1 is greater than the sum of the bid weights of the B_2, then the objective value z_1 is always greater than the objective value z_2. Therefore, we will be proving that for any two feasible solutions, the solution with the higher sum of the bid weights has higher objective value independent of the sum of the weights of the requested courses.

Suppose that the sum of the bid weights of the B_1 is greater than the sum of the bid weights of the B_2,

$$\sum_{\forall i \mid b_i \in B_1} w_i > \sum_{\forall i \mid b_i \in B_2} w_i. \tag{11}$$

The difference between the sums of the bid weights of B_1 and B_2 is

$$\left(\sum_{\forall i \mid b_i \in B_1} w_i - \sum_{\forall i \mid b_i \in B_2} w_i \right) = 2^k \quad (k \in \mathbb{Z}^+ \cup \{0\}). \tag{12}$$

Then, the difference between the objective values z_1 and z_2 is

$$z_1 - z_2 = 2^k + \alpha \cdot \left(\sum_{\forall i \mid b_i \in B_1} \sum_{\forall j \mid (a_{ij}, w_{ij}) \in R_i \wedge y_{ij} = 1} w_{ij} - \sum_{\forall i \mid b_i \in B_2} \sum_{\forall j \mid (a_{ij}, w_{ij}) \in R_i \wedge y_{ij} = 1} w_{ij} \right). \tag{13}$$

Since exactly one requested course is added for each satisfied bid, the lower bound for the sum of the requested course weights of B_1 is

$$\sum_{\forall i \mid b_i \in B_1} \sum_{\forall j \mid (a_{ij}, w_{ij}) \in R_i \wedge y_{ij} = 1} w_{ij} \geq \min_{i,j} w_{ij} = 1. \tag{14}$$

Because of the conditional proof assumption in Eq. (11), B_1 cannot be a subset of B_2, and therefore the upper bound for the sum of the requested course weights of B_2 is

$$\sum_{\forall i|b_i\in B_2}\ \sum_{\forall j|(a_{ij},w_{ij})\in R_i\wedge y_{ij}=1} w_{ij} \le (|B|-1)\cdot \max_{i,j} w_{ij} = (|B|-1)\cdot m. \tag{15}$$

Therefore,

$$\left(\sum_{\forall i|b_i\in B_1}\ \sum_{\forall j|(a_{ij},w_{ij})\in R_i\wedge y_{ij}=1} w_{ij} - \sum_{\forall i|b_i\in B_2}\ \sum_{\forall j|(a_{ij},w_{ij})\in R_i\wedge y_{ij}=1} w_{ij}\right) \ge 1-(|B|-1)\cdot m. \tag{16}$$

Using $\alpha = 1/(|B|-1)\cdot m$, the smallest difference between the objective values is:

$$z_1 - z_2 \ge 2^0 + \frac{1}{(|B|-1)\cdot m}\cdot[1-(|B|-1)\cdot m] \tag{17}$$

$$z_1 - z_2 \ge \frac{1}{(|B|-1)\cdot m} \tag{18}$$

$$z_1 - z_2 \ge 0. \tag{19}$$

Therefore, the solution with the higher sum of the bid weights has higher objective value independent of the sum of the weights of the requested courses. □

In general, as the number of courses with remaining quotas increases, the number of solutions with identical values in the first summand of the objective function is likely to increase. The reason is that for satisfiable bids there will be more than one alternative requested course that can be added. Hence, the weight mechanism for the requested courses and α factor mechanism will play an important role for increasing the satisfaction of students in these cases by enabling their favored courses to be added to their schedule.

4. Solution procedure

Since the direct barter model can be formulated using integer programming, its problem instances can be solved using general purpose integer programming solvers. However, resemblance of this model to the used car salesman problem (UCSP) in [22] and the polynomial time barter models in [23,24] motivated us to search for a network flow based solution. Because of the bid weights and the recursive definition of bid satisfiability that causes circular patterns in the solution like UCSP, we modeled the direct barter problem as a minimum cost flow problem [1]. The minimum cost flow problem is defined as follows: let $N(V, A, l, u, c, b)$ denote a network with node set V, arc set A, lower bound $l(v, w)$, capacity $u(v, w)$, cost $c(v, w)$ values for each arc $(v, w) \in A$, and supply/demand values $b(v)$ for each node $v \in V$. Let $x(v, w)$ represent the flow on arc $(v, w) \in A$. The minimum cost flow problem is defined as follows:

$$\text{Minimize} \quad \sum_{\forall v,w|(v,w)\in A} c(v,w)\cdot x(v,w) \tag{20}$$

$$\text{s.t.} \quad \sum_{\forall w|(v,w)\in A} x(v,w) - \sum_{\forall w|(w,v)\in A} x(w,v) = b(v)\ (\forall v \mid v\in V) \tag{21}$$

$$l(v,w) \le x(v,w) \le u(v,w) \quad (\forall v, w \mid (v,w)\in A) \tag{22}$$

where $\sum_{\forall v|v\in V} b(v) = 0$.

To help us in defining the network in our problem formally, we first introduce a set P, called *restriction-pairs set*, which consists of course–student pairs. The set P is defined as follows:

$$P = \{(c_k, s_l) | c_k \in C \wedge s_l \in S \wedge (\exists i, i', j, j' \mid i \ne i' \wedge b_i, b_{i'} \in B_l \wedge (c_k, w_{ij}) \in R_i \wedge (c_k, w_{i'j'}) \in R_{i'})\}.$$

Thus, each pair (c_k, s_l) in P indicates that the student s_l requests the course c_k in his at least two different bids, and therefore the student s_l must be prevented from adding the course c_k more than once in the final solution. Based on this definition, the minimum cost flow network can be constructed as follows:

- The set of nodes, V, consists of four types of nodes:
  - (i) a course node c_k for each course $c_k \in C$,
  - (ii) a special node CENTER that represents the null course to be dropped for add bids,
  - (iii) a bid node b_i for each bid $b_i \in B$,
  - (iv) a restriction node r_{kl} for each $(c_k, s_l) \in P$ for preventing the student s_l from adding the course c_k more than once.

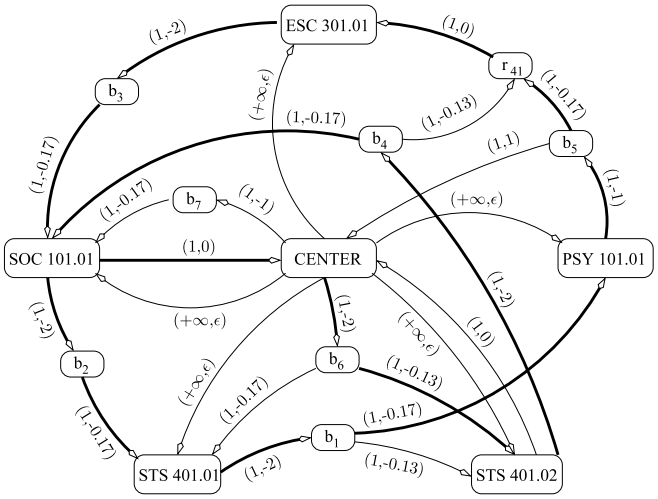

Fig. 2. Minimum cost flow network of the example given in Section 2 with (capacity, cost) values on the arcs. The solution is shown with the bold arcs where one unit of flow passes in each bold arc.

- The set of arcs, A, consists of seven types of arcs:
  - (i) an arc (c_k, CENTER) for each course $c_k \in C$ with capacity equal to q_{c_k} and cost equal to 0 which represents the remaining quota of the course c_k,
  - (ii) an arc (CENTER, c_k) for each course $c_k \in C$ with capacity equal to $+\infty$ and cost equal to ϵ,
  - (iii) an arc (d_i, b_i) for each *barter* and *drop-unless-barter* bid $b_i = (d_i, w_i, R_i)$ with capacity equal to 1 and cost equal to $-w_i$,
  - (iv) an arc (CENTER, b_i) for each *add* bid $b_i = (c_\emptyset, w_i, R_i)$ with capacity equal to 1 and cost equal to $-w_i$,
  - (v) for each course $c_k \in C$ and for each student $s_l \in S$ such that $(c_k, s_l) \in P$:
    - (a) an arc (b_i, r_{kl}) for each bid $b_i = (d_i, w_i, R_i) \in B_l$ if there exists a tuple $(c_k, w_{ij}) \in R_i$ with capacity equal to 1 and cost equal to $-\alpha \cdot w_{ij}$, (i.e. $\forall i, j, k, l \mid s_l \in S \wedge c_k \in C \wedge b_i \in B_l \wedge (a_{ij}, w_{ij}) \in R_i \wedge c_k = a_{ij} \wedge (c_k, s_l) \in P$: an arc (b_i, r_{kl})),
    - (b) an arc (r_{kl}, c_k) with capacity equal to 1 and cost equal to 0, (i.e. $\forall k, l \mid s_l \in S \wedge c_k \in C \wedge (c_k, s_l) \in P$: an arc (r_{kl}, c_k)),
  - (vi) for each course $c_k \in C$ and for each student $s_l \in S$ such that $(c_k, s_l) \notin P$:
    - (a) an arc (b_i, c_k) for each bid $b_i = (d_i, w_i, R_i) \in B_l$ if there exists a tuple $(c_k, w_{ij}) \in R_i$ with capacity equal to 1 and cost equal to $-\alpha \cdot w_{ij}$, (i.e. $\forall i, j, k, l \mid s_l \in S \wedge c_k \in C \wedge b_i \in B_l \wedge (a_{ij}, w_{ij}) \in R_i \wedge c_k = a_{ij} \wedge (c_k, s_l) \notin P$: an arc (b_i, c_k))
  - (vii) an arc (b_i, CENTER) for each drop-unless-barter bid $b_i \in D$ with capacity equal to 1 and cost equal to w_i.

Lower bounds $l(v, w)$ for all arcs $(v, w) \in A$ are set to 0. Similarly, there is no supply or demand for any node in the network, and therefore $b(v) = 0$ for every node $v \in V$. Note that ϵ which is used as the cost of the arcs of type (ii) is the smallest possible positive number representable on the computer. It is used to prevent zero cost cycles.

The minimum cost flow network for the example problem given in Section 2 and its solution can be seen in Fig. 2. As stated earlier, all drop bids should always be satisfied and since they are always part of the solution, they need not to be included in the network. Therefore, before constructing the network, as a preprocessing step all drop bids are marked as satisfied and the remaining quotas of the courses are increased accordingly. For instance, if there are z drop bids for the course c_k, after satisfying these bids the remaining quota of the course becomes $q_{c_k} + z$. So, when presenting the network, we assume that there will be no drop bids in the bid set B and the set of remaining quotas Q is adjusted accordingly. This simple preprocessing step eliminates drop bids and reduces the network size. Therefore, for the example problem, the drop bid 8 is marked as satisfied beforehand and the quota of the course SOC 101.01 is increased by one.

Verifying the correctness of the described network is straightforward. The arcs of types (iii) and (iv) represent the binary decision variables x_i for barter and add bids respectively. Therefore, additive inverses of the bid weights, $-w_i$, are used as the costs of these arcs. This statement is also valid for the drop-unless-barter bids on the condition that there is no flow on the corresponding arc of type (vii). However, if there is a flow passing through both the arc of type (iii) and the arc of type (vii) for a drop-unless-barter bid, this means that only the drop part of the bid is satisfied. In this case, this bid is considered as unsatisfied in accordance with the IP formulation in Section 3 and the sum of the costs of the corresponding arcs of type (iii) and (vii) are zero. As in the case of the bid weights, additive inverses of the weights of the requested courses, $-\alpha \cdot w_{ij}$, are used as the costs of type (v.a) and (vi.a) arcs that represent binary decision variables y_{ij}. Costs of the other arcs are zero. Therefore, minimization of the cost yields maximization of the sum of the weights. In order to satisfy a bid, one unit of flow must flow from the node representing the course to be dropped (for add bids, from the center node) to the node representing

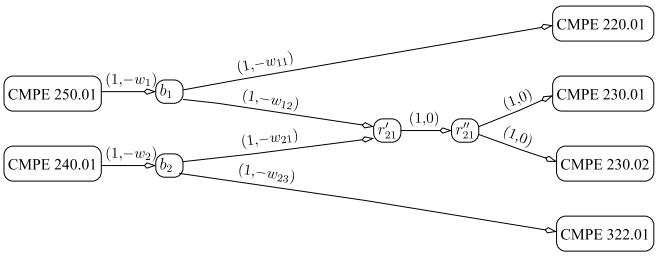

Fig. 3. Network for illustrating the usage of the restriction nodes for a course restriction set with (capacity, cost) values on the arcs.

the course to be added. The capacity limits on type (iii) and (iv) arcs ensure that only one of the requested courses is added for each satisfied bid. Also, the capacity limits on type (v.b) arcs prevent students from registering for a course more than once. The arcs of type (i) represent the remaining quotas of the respective courses and type (ii) arcs allow satisfaction of barter bids when the courses to be dropped are not requested by any other satisfied bid. Finally, the arcs of type (vii) allow barter bids which are marked as drop-unless-barter to drop the course if the bartering is not possible. Quota restrictions of the courses are enforced using the flow conservation property of the network nodes c_k that correspond to the courses. Outgoing flow from a course node is restricted with the remaining quota (adjusted value according to the drop bids in the preprocessing step) plus the number of satisfied bids that drop the course.

When the minimum cost flow is found on the network, winning bids can be determined by checking flow on arc types (iii) and (iv) for barter and add bids respectively. If the flow on an arc of these types is 1, it shows that corresponding barter or add bid is satisfied and it is in the optimum solution. Similarly, the arc of types (v.a) or (vi.a) that originates from the winning bid determines the course to be added for that bid. Since there can only be one arc with 1 unit of flow among these arcs, the head of this arc shows the course to be added for the winning bid.

There are strongly polynomial algorithms for solving minimum cost network flow problems such as the minimum mean cycle-canceling algorithm with time complexity $O(|V|^2|A|^3 \log |V|)$ [13] and the enhanced capacity scaling algorithm with time complexity $O((|A| \log |V|)(|A| + |V| \log |V|))$ [1]. Since the minimum cost flow network for a direct barter problem instance can be constructed in polynomial time using the above algorithm, the optimum solution of the direct barter problem can also be found in polynomial time and hence the IP formulation given in Section 3 is in P.

Extending the functionality of the restriction nodes

The function of the restriction nodes in the network is to prevent a student from registering the same course more than once. However, the function of these nodes can be extended so that instead of a single course, any number of disjoint course restriction sets can be defined for each student such that the student can register for at most one of the courses in this set. This is especially useful when a student requests more than one section of the same course or a set of conflicting courses in his at least two bids. For instance, assume that a student submits the following bids:

$$\texttt{CMPE 250.01} \xrightarrow{w_1} \{(\texttt{CMPE 220.01}, w_{11}), (\texttt{CMPE 230.01}, w_{12}), (\texttt{CMPE 230.02}, w_{13})\}$$
$$\texttt{CMPE 240.01} \xrightarrow{w_2} \{(\texttt{CMPE 230.01}, w_{21}), (\texttt{CMPE 230.02}, w_{22}), (\texttt{CMPE 322.01}, w_{23})\}$$

It is clear that the student cannot register for two different sections of CMPE 230 at the same time. Therefore, in order enforce this restriction, we define a course restriction set that consists of restricted courses CMPE 230.01 and 230.02. Instead of using separate restriction nodes for these restricted courses, a split restriction node pair is introduced for this set as shown in Fig. 3. Then, for each bid of the student that requests at least one of the courses in this set, an arc is drawn from the corresponding bid node to the first node of the pair. The capacity of this arc is set to 1 and the cost of this arc is set to the additive inverse of the weight of the highest ranked restricted course in the bid. Additionally, an arc is drawn between the pair nodes with one unit of capacity and zero cost. This arc limits the number of restricted courses to be added to one. Finally, for each restricted course, an arc is drawn from the second node of the pair to the corresponding course node with a capacity of 1 and cost of 0. This procedure is repeated by introducing a restriction node pair for each course restriction set defined.

5. Experimental results

In order to estimate the real-world performance and the quality of the solutions of our barter model, we developed a test case generator based on real-world student registration data obtained from the Boğaziçi University registration system [5] for the academic year 2008–2009. Since the number of students in our university is relatively small, that is 7095, we generated a statistical profile using the actual registration data and determined the parameters of the test case generator accordingly so that it is capable of generating test cases for arbitrary number of students. The parameters of the test case generator and its source code can be found in [28].

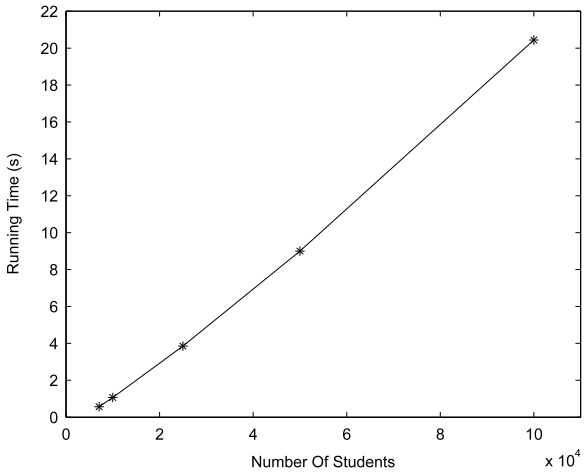

Fig. 4. Graph of the number of students vs. running time (seconds) of the network solver.

Table 1
Running times of the network solver for the test cases (seconds).

# Students	# Courses	Avg. # Bids	Running time (s)	
			mean	stdev
[a] 7095	1 158	22,700	0.57	0.03
10,000	1 632	32,090	1.07	0.04
25,000	4 080	80,012	3.85	0.13
50,000	8 160	160,099	9.00	0.29
100,000	16,321	320,675	20.44	0.75

[a] This row presents the results for Boğaziçi University.

We conducted two different experiments on a dedicated 64 bit Intel Xeon 2.66 GHz workstation with 8 GB memory using Linux operating system. We used CS2 software which contains a solver for the minimum cost flow problems based on scaling push–relabel algorithm [8,12].

In the first experiment, a group of 20 test cases are generated for each selected number of students ranging between 7095 and 100,000. The average running time of the network solver for each group and the corresponding standard deviation are presented in Table 1 and the associated plot is depicted in Fig. 4. As seen from the results, the solver finds the solution of the problem instances with 100,000 students and approximately 320,000 bids in less than 21 s which is quite small. For the case of our university, on the other hand, each instance is solved in less than one second.

In the second experiment, the solutions of our barter model are compared with the currently used FCFS based system. The purpose of this experiment is to present the improvement in the optimum solutions of the test cases over the FCFS approach under different occupancy rates for the courses. Thus, the importance of the introduced bartering mechanism and the weighting mechanism could be observed. In order to simulate the FCFS system, a random permutation of the bids in each test case is generated by preserving the preferred order of bids of each student. The bids in the permuted list are processed one by one, simulating the way the students submit the bids to the registration system. The processing step is straightforward; for each bid, the remaining quotas of the courses in the request set is checked in the order of students' preferences and if one empty slot is found, the course is added to the schedule of the student. If the processed bid is a barter bid, then the course to be dropped is also removed from his schedule and the remaining quota of that course is increased by one. The whole simulation process is repeated up to five times for the unsatisfied bids.

The second experiment is conducted for two different numbers of students, that is 7095 and 50,000, where the number of courses are 1158 and 8160 respectively. For each number of students, the ratio of the courses without remaining quota to the number of all courses, called p, is varied between 0.20 and 0.99 meaning that approximately $(100 \cdot p)\%$ of the courses have no remaining quota. For each configuration, again a group of 20 test cases are generated. The results of this experiment are given in Table 2.

For the test cases with 7095 students, assuming that 20% of the courses are full, the number of satisfied bids in the solution found by the barter model is approximately 12% higher than that of the FCFS based system. For the case of our university where approximately 28% of the courses are full, the barter model provides 15% better results. It is remarkable that the improvement percentage for the number of satisfied bids increases exponentially as p increases. In the extreme situation where the courses are 99% full, approximately 2800% improvement over the FCFS system is observed. It should also be noted that in the barter model the standard deviations are also very small relative to the mean values so that the quality of the solutions found do not vary much. The results of the test cases with 50,000 students are also very close to the test cases with

Table 2
Improvements in the solutions of the barter model over the FCFS model.

# Stu.	(p)	# Bids	# Satisfied bids (mean/stdev)			# Satisfied students (mean/stdev)		
			FCFS	Barter	Impr. (%)	FCFS	Barter	Impr. (%)
7095	0.20	22,757	18,367/148	20,642/129	**12.4**	6719/17	6828/15	**1.6**
	[a]0.28	22,725	17,823/176	20,409/153	**14.5**	6674/17	6822/16	**2.2**
	0.40	22,685	16,847/184	19,905/155	**18.2**	6605/23	6820/12	**3.2**
	0.60	22,761	14,224/260	18,656/140	**31.2**	6322/39	6814/14	**7.8**
	0.80	22,729	9174/437	17,212/190	**88.0**	5329/128	6812/16	**27.9**
	0.90	22,723	5103/497	16,300/116	**224**	3737/267	6760/23	**82**
	0.95	22,745	2660/503	16,050/163	**523**	2249/356	6671/24	**203**
	0.99	22,726	556/123	15,569/151	**2827**	537/114	6580/25	**1178**
50,000	0.20	160,226	129,495/513	145,312/405	**12.2**	47,285/53	48,040/47	**1.6**
	0.40	160,312	118,538/595	140,396/407	**18.4**	46,469/68	48,033/37	**3.4**
	0.60	160,319	100,492/1138	131,704/506	**31.1**	44,625/167	48,039/41	**7.7**
	0.80	160,274	64,312/1330	121,217/416	**88.6**	37,416/383	48,017/55	**28.3**
	0.90	160,052	35,624/1321	115,162/326	**223**	26,203/691	47,675/57	**82**
	0.95	160,477	18,984/994	113,228/345	**498**	16,084/731	47,010/72	**193**
	0.99	160,276	3597/507	109,539/298	**3003**	3480/476	46,300/43	**1254**

[a] This row presents the results for Boğaziçi University.

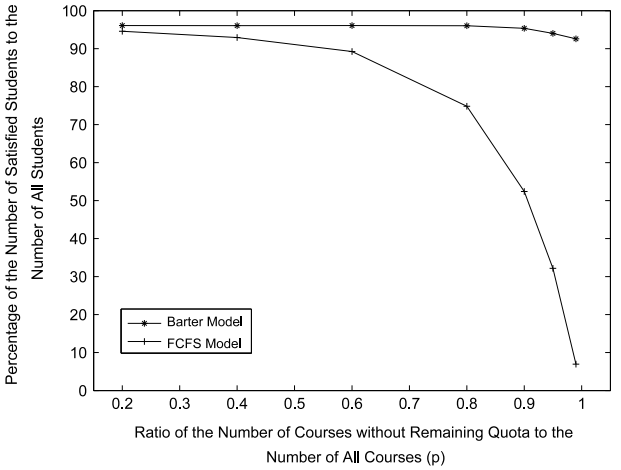

Fig. 5. Graph comparing the number of satisfied students in the solutions of the barter model and the FCFS model.

7095 students showing that the model is scalable to the universities with higher number of students without sacrificing the quality of the solutions.

By virtue of the weighting mechanism, the barter model also improves fairness among the students by increasing not only the number of satisfied bids but also the number of satisfied students. As seen from the plot in Fig. 5, the number of satisfied students in the barter model whose at least one bid is satisfied is approximately 96% of the total number of students and decreases slightly to 93% as p goes to 0.99. However, in the FCFS model, starting from 95%, this ratio drops to 7%.

6. Previous work

As stated earlier, the CT and SS problems have been covered extensively in the literature. Some studies have addressed solely the CT problem; several integer linear programming [3,9,18] models and heuristic methods [3,9,15] have been proposed. For the SS problem, Reeves and Hickman [25] and Willoughby and Zappe [30] have proposed a mixed integer and network programming models respectively, and Alvarez-Valdes et al. [2] have applied Tabu search techniques. In [16,26], on the other hand, unified optimization models that address both problems are presented. There are also open source (for example [29]) and commercial packages (for example [17]) that address these problems. For further details on these problems, the reader is referred to surveys by Burke and Petrovic [6], Carter and Laporte [7], Lewis [20] and Schaerf [27].

In the remaining of this section, we first review Graves et al.'s work [14] in detail which has addressed the student course add/drop process as we do in this paper. Then, in Section 6.2, we discuss the relationship between our model and a somewhat related problem called the stable college admission problem (SCAP) [10,4].

6.1. Relationship between Graves et al.'s work and the barter model

Graves et al. [14] propose an auction based market approach complete with clearing prices for allocating course sections to students. Their model consists of two rounds. In the first registration round, which is called registration bidding system (RBS), students are granted bidding points (i.e. registration money) which they can use to bid on desired schedules. During this period, students are allowed to place course selections as their bids together with the money they will pay for each schedule. The bids are ranked in descending order of bidding points and are selected if requested course capacities are available. At the end of the registration period, the prices of the courses are determined. Each successful bidder pays the sum of the prices of the assigned courses to him instead of the price he offered for his bid. Therefore, it is possible that a successful bidder may not have enough money in which case a subsidy given by the system covers the deficiency. Subsidies not paid back during add/drop phase as a result of dropping courses are simply forgotten (waived) by the system. Hence, we believe that if this fact is known by the students, then it could easily be abused by offering high prices for their schedules. Since the winning bids can be subsidized and the subsidized amounts can be forgotten, this then introduces fairness problems in Graves et al.'s approach.

In the second round, which is called drop/add/swap (DAS) round, Graves et al. introduce a course swapping idea. This corresponds to the barter scheme that we propose in this paper. As in the RBS round, an auction based approach is used. Students submit add, drop and swap bids together with the amounts of bidding points that are carried forward from the RBS round (if any). After the end of the round, a linear program whose objective is to maximize the sum of the bidding points of the satisfied bids is solved. By solving the linear program, the students are assigned to the courses and also the prices of the courses in terms of bidding points (dual prices) are determined. As in the registration round, each student pays the actual amounts of his satisfied bids calculated according to the determined prices of the courses. However, in this case it is stated that subsidies are not allowed. Since there is no formal treatment for this problem except for a simple linear programming example, it is not clear how the restriction for preventing the subsidies is applied in their model. This also prevents one from implementing their model. Although, by the definition of dual prices, the actual amount to be paid for each satisfied bid is bounded by the offered price for that bid, this does not solve the subsidy problem since each student may submit more than one bid. Thus, a further constraint is necessary. In fact, the following proposition shows that introducing a budget constraint in order to prevent subsidies, i.e. the sum of the bidding points of the satisfied bids of a student should be less than or equal to the amount of bidding points owned by that student, to the given linear programming example makes the resulting problem NP-hard. It should be noted that the formal definition of the DAS problem given below is constructed by us according to the linear programming example and the explanations given in [14].

Proposition 2. *The decision version of the DAS problem with budget constraint is NP-complete.*

Proof. Let Π be the decision version of the DAS problem with budget constraint. Π is defined as follows: given

- a set of courses, $C = \{c_1, c_2, \ldots, c_m\}$;
- a sequence of remaining quotas, $Q = \{q_{c_1}, q_{c_2}, \ldots, q_{c_m}\}$ where q_{c_k} is the remaining quota of course c_k $(1 \le k \le m,\ q_{c_k} \in \mathbb{Z}^+ \cup \{0\})$;
- a set of students, $S = \{s_1, s_2, \ldots, s_t\}$;
- a set of bids, $B = \bigcup_{l=1}^{t} B_l$ where B_l is the set of bids of a student s_l $(1 \le l \le t)$ and each bid is denoted by a triplet, $b_i = (d_i, a_i, p_i)$, where d_i is the course to be dropped for *barter* and *drop* bids or the null course, $c_\emptyset$, for *add* bids $(d_i \in C \cup \{c_\emptyset\})$, a_i is the course to be added for *barter* and *add* bids or the null course, $c_\emptyset$, for *drop* bids $(a_i \in C \cup \{c_\emptyset\})$, and p_i is the amount of bidding points offered by the student for bid $b_i (1 \le i \le n = |B|, p_i \in \mathbb{Z}^+ \cup \{0\})$;
- a set of bid restrictions, $L = \{l_1, l_2, \ldots, l_z\}$ that consists of mutually disjoint subsets of bids, $l_y \subseteq B$ $(1 \le y \le z)$, such that at most one of the bids in l_y can be satisfied (e.g. a student may put a restriction on two of his add bids so that only one of them can be satisfied or the system may enforce a restriction on two barter bids of a student in which the same course is dropped);
- a sequence of bidding points owned by students, $F = \{f_1, f_2, \ldots, f_t\}$ where f_l is the amount of bidding points owned by student s_l $(1 \le l \le t, f_l \in \mathbb{Z}^+ \cup \{0\})$;
- a positive integer K;

is there a subset $B' \subseteq B$ such that the following inequalities are satisfied?

$$\sum_{\forall i | b_i \in B'} p_i \ge K \tag{23}$$

$$\sum_{\forall i | b_i \in (B_l \cap B')} p_i \le f_l \quad (\forall l \mid s_l \in S) \tag{24}$$

$$\sum_{\forall i | b_i \in B' \wedge a_i = c_k} 1 - \sum_{\forall i | b_i \in B' \wedge d_i = c_k} 1 \le q_{c_k} \quad (\forall k \mid c_k \in C) \tag{25}$$

$$\sum_{\forall i | b_i \in (l_y \cap B')} 1 \le 1 \quad (\forall y \mid l_y \in L). \tag{26}$$

In this formulation, Eq. (23) ensures that the sum of the offered bidding points for all satisfied bids (B') is greater than or equal to the positive integer K. Eq. (24) is the budget constraint which prevents the sum of the bidding points of the satisfied bids of a student from exceeding the amount of bidding points owned by that student. The quota restrictions are enforced in Eq. (25). For each course, the number of students who drop the course plus the remaining quota of the course should be greater than or equal to the number of students who add the course. Finally, Eq. (26) ensures that bid restrictions are applied such that for all y ($1 \leq y \leq z$), at most one of the bids in $l_y \in L$ is satisfied.

If we have a certificate that consists of $B' \subseteq B$, this certificate can be verified in polynomial time by checking Eqs. (23)–(26). Therefore Π is in NP.

Next, we present a polynomial time transformation from the subset sum problem. Let Π' be the subset sum problem (see for example: [19, p. 73] and [11, p. 247]) which is defined as follows: given a finite set U, a weight value $w(u_i) \in \mathbb{Z}^+$ for each $u_i \in U$ ($1 \leq i \leq |U|$), and positive integers C and K, is there a subset $U' \subseteq U$ such that

$$\sum_{\forall i | u_i \in U'} w(u_i) \geq K \tag{27}$$

$$\sum_{\forall i | u_i \in U'} w(u_i) \leq C. \tag{28}$$

Let $\Pi'(U, w(u_i), C, K)$ be an instance of the subset sum problem. It can be transformed to Π in polynomial time as follows: let the set of courses C consist of $|U|$ courses ($C = \{c_1, c_2, \ldots, c_{|U|}\}$) and the remaining quotas of all the courses be 1 ($q_{c_k} = 1,\ k = 1, 2, \ldots, |U|$). The students set S consists of 1 student ($S = \{s_1\}$) and for each $u_i \in U$, the student s_1 submits an add bid b_i requesting the course c_i with a price of $w(u_i)$ ($i = 1, 2, \ldots, |U|$). The amount of bidding points of the student s_1 is C ($F = \{C\}$). Let the set of bid restrictions, L, be empty. Since in each bid exactly one unique course is requested and the remaining quotas of all the courses are 1, Eq. (25) always holds independent of the set B'. Thus, for the transformed problem instances, Eqs. (23)–(26) reduce to the inequalities of the subset sum problem:

$$\sum_{\forall i | b_i \in B'} w(u_i) \geq K \tag{29}$$

$$\sum_{\forall i | b_i \in B'} w(u_i) \leq C \tag{30}$$

and therefore, the solution of a problem instance of Π is also the solution of the corresponding problem instance of Π', and the solution of a problem instance of Π' is also the solution of the problem instance of Π.

Since Π is in NP and the subset sum problem is NP-complete, the decision version of the DAS problem with dynamic credit constraint is NP-complete. □

Besides the subsidy and the associated unfairness problems in Graves et al. approach, students are also responsible for deciding the prices of the bids according to certain upper and lower bounds. However, determining the prices can be cumbersome for the students because of the combinatorial nature of the model. Furthermore, since the remaining points of the students are transferred to the next semester, decision making will become tougher since the students should also consider the following semesters. Finally, we note that although the students' perception of the quality of their schedules is not quantified, Graves et al. estimate one percent increase in the quality of schedules using their model.

6.2. Relationship with the stable university admission problem

The preference based ordering of bids and requested courses indicates a relationship between the direct barter model and the stable college admission problem (SCAP) [10,4]. In the SCAP, there are two sets of agents, colleges and students. Each college has strict preferences over the students and can accept a limited number of students. Each student, on the other hand, can enroll to only one college and has also strict preferences over the colleges. The SCAP is defined as finding a matching of students to colleges, called a *stable allocation*, such that no unmatched pair of opposite agents would simultaneously be better off if they were matched together.

In general, the SCAP cannot be used to solve our barter model. However, if we consider a simple special case of our model in which *only add bids are allowed* and *the weights of the bids are unique*, then this simple problem can be reduced to the SCAP as follows[1]: let the set of bids B and the set of courses C in our barter model map to the row agents I (students), and the column agents J (colleges) in [4] respectively. The function $\pi(i, j)$ in the SCAP indicates whether student i wants to admit to college j or not. Therefore, for each bid $b_i \in B$ and for each course $c_i \in C$ in our model, if c_i is requested in bid b_i, then $\pi(i, j)$ is set to 1. Otherwise, it is set to 0. For each bid $b_i \in B$ in our model, the weights of the requested courses correspond to the strict college preference of the student i. For each course $c_k \in C$ in our model, the weights of the bids that request

[1] Note that the latter requirement cannot be satisfied when the bid weight function in Eq. (7) is used. Therefore, even the instances of direct barter model consisting of add bids cannot be reduced to the SCAP.

the course c_k correspond to the strict student preference of the college j. In the SCAP, each student i can enroll at most $s(i)$ colleges and each college j can accept $d(j)$ students. In our model, since only one course can be assigned to a bid b_i, we set $s(i) = 1$. However, a course c_k can be assigned to q_{c_k} students, and therefore we set $d(j) = q_{c_k}$. Then, the "college optimal" deferred acceptance procedure [10] produces a stable allocation in which the courses are assigned to the bids in such a way that the bid weights are favored against the weights of the requested courses. However, the stable allocation found by the deferred acceptance procedure can be different from the allocation found by the direct barter model since the direct barter model does not seek a stable allocation but an allocation that maximizes total satisfaction of students. For instance, assume that the following two add bids are submitted:

Student 1 (bid 1): $c_\emptyset \rightarrow$ {course A, course B}
Student 2 (bid 2): $c_\emptyset \rightarrow$ {course A}

Suppose that the weight of the first bid is higher than that of the second bid. Then, the deferred acceptance procedure would assign course A to Student 1 which is the only stable allocation. However, using the direct barter model both bids would be satisfied and Student 1 would get course B and Student 2 would get course A.

7. Conclusion

In this paper, we have modeled the course add/drop process as a direct bartering problem in which add/drop requests appear as bids. We formulated the resulting problem as an integer linear program, and then showed that our problem can be solved in polynomial time as a minimum cost network flow problem. In our model, we also introduced a two-level weighting mechanism that enables students to express their preferences for their bids and for the requested courses. The weighting mechanism also improves fairness among the students. As demonstrated with the experimental results, the minimum cost flow network solvers can solve problem instances for a typical university within seconds. Hence, our algorithms can be deployed in universities with hundreds of thousands of students without worrying about execution times. The experimental results also show significant improvement in the quality of the solutions over the currently used FCFS based system while preserving the fairness, and hence we believe that the direct bartering model will greatly improve the satisfaction of students in the universities.

As a final note, the real-life performance of our problem also motivates us to apply the direct barter approach to different application areas. For instance, an electronic exchange that facilitates bartering e-media such as e-books, music and movies on the Internet can be designed in a similar manner.

References

[1] R.K. Ahuja, T.L. Magnanti, J.B. Orlin, Network Flows: Theory, Algorithms and Applications, Prentice Hall, New Jersey, NJ, USA, 1993.
[2] R. Alvarez-Valdes, E. Crespo, J.M. Tamarit, Assigning students to course sections using tabu search, Annals of Operations Research 96 (1–4) (2000) 1–16.
[3] S.M. Al-Yakoob, H.D. Sherali, Mathematical programming models and algorithms for a class-faculty assignment problem, European Journal of Operational Research 173 (2) (2006) 488–507.
[4] M. Baïou, M. Balinski, The stable allocation (or ordinal transportation) problem, Mathematics of Operations Research 27 (4) (2002) 662–680 (erratum).
[5] Boğaziçi university online registration system, 2010. URL: http://registration.boun.edu.tr/ (accessed on 18.04.10).
[6] E.K. Burke, S. Petrovic, Recent research directions in automated timetabling, European Journal of Operational Research 140 (2) (2002) 266–280.
[7] M.W. Carter, G. Laporte, Recent developments in practical course timetabling, in: Practice and Theory of Automated Timetabling II, in: Lecture Notes in Computer Science, vol. 1408, Springer-Verlag, Berlin, 1998, pp. 3–19.
[8] Andrew Goldberg's network optimization library, 2010. URL: http://www.avglab.com/andrew/soft.html (accessed on 18.04.10).
[9] M. Dimopoulou, P. Miliotis, Implementation of a university course and examination timetabling system, European Journal of Operational Research 130 (1) (2001) 202–213.
[10] D. Gale, L.S. Shapley, College admissions and the stability of marriage, The American Mathematical Monthly 69 (1) (1962) 9–15.
[11] M.R. Garey, D.S. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness, WH Freeman and Co., San Francisco, CA, USA, 1979.
[12] A.V. Goldberg, An efficient implementation of a scaling minimum-cost flow algorithm, Journal of Algorithms 22 (1) (1997) 1–29.
[13] A.V. Goldberg, R.E. Tarjan, Finding minimum-cost circulations by canceling negative cycles, Journal of the ACM 36 (4) (1989) 873–886.
[14] R.L. Graves, L. Schrage, J. Sankaran, An auction method for course registration, Interfaces 23 (5) (1993) 81–92.
[15] A. Hertz, Finding a feasible course schedule using tabu search, Discrete Applied Mathematics 35 (3) (1992) 255–270.
[16] A. Hertz, V. Robert, Constructing a course schedule by solving a series of assignment type problems, European Journal of Operational Research 108 (3) (1998) 585–603.
[17] T.R. Hinkin, G.M. Thompson, Schedulexpert: scheduling courses in the Cornell University school of hotel administration, Interfaces 32 (6) (2002) 45–57.
[18] N.A. Ismayilova, M. Sağır, R.N. Gasimov, A multiobjective faculty-course-time slot assignment problem with preferences, Mathematical and Computer Modelling 46 (7–8) (2007) 1017–1029.
[19] H. Kellerer, U. Pferschy, D. Pisinger, Knapsack Problems, Springer, Berlin, Germany, 2004.
[20] R. Lewis, A survey of metaheuristic-based techniques for university timetabling problems, OR Spectrum 30 (2008) 167–190.
[21] T. Müller, K. Murray, Comprehensive approach to student sectioning, in: Proceedings of the 7th International Conference on the Practice and Theory of Automated Timetabling, 2008.
[22] C. Özturan, Used car salesman problem: a differential auction–barter market, Annals of Mathematics and Artificial Intelligence 44 (3) (2005) 255–267.
[23] C. Özturan, Resource bartering in data grids, Scientific Programming 12 (3) (2004) 155–168.
[24] C. Özturan, Network flow models for electronic barter exchanges, Journal of Organizational Computing and Electronic Commerce 14 (3) (2004) 175–194.
[25] G.R. Reeves, E.P. Hickman, Assigning MBA students to field-study project teams—a multicriteria approach, Interfaces 22 (5) (1992) 52–58.
[26] S.E. Sampson, J.R. Freeland, E.N. Weiss, Class scheduling to maximize participant satisfaction, Interfaces 25 (3) (1995) 30–41.

[27] A. Schaerf, A survey of automated timetabling, Artificial Intelligence Review 13 (2) (1999) 87–127.
[28] Software and supplementary materials for 'a direct barter model for course add/drop process', 2010. URL: http://www.ahozer.com/en/pubs/adddrop/ (accessed on 18.04.10).
[29] UniTime, 2010. URL: http://www.unitime.org/ (accessed on 18.04.10).
[30] K.A. Willoughby, C.J. Zappe, A methodology to optimize foundation seminar assignments, Journal of the Operational Research Society 57 (8) (2006) 950–956.